

Background modeling by shifted tilings of stacked denoising autoencoders

Jorge García-González, Juan M. Ortiz-de-Lazcano-Lobato, Rafael M. Luque-Baena, and Ezequiel López-Rubio

Department of Computer Languages and Computer Sciences. University of Málaga.
Bulevar Louis Pasteur, 35. 29071 Málaga. Spain.
{jorgegarcia,jmortiz,rmluque,ezeqlr}@lcc.uma.es

Abstract. The effective processing of visual data without interruption is currently of supreme importance. For that purpose, the analysis system must adapt to events that may affect the data quality and maintain its performance level over time. A methodology for background modeling and foreground detection, whose main characteristic is its robustness against stationary noise, is presented in the paper. The system is based on a stacked denoising autoencoder which extracts a set of significant features for each patch of several shifted tilings of the video frame. A probabilistic model for each patch is learned. The distinct patches which include a particular pixel are considered for that pixel classification. The experiments show that classical methods existing in the literature experience drastic performance drops when noise is present in the video sequences, whereas the proposed one seems to be slightly affected. This fact corroborates the idea of robustness of our proposal, in addition to its usefulness for the processing and analysis of continuous data during uninterrupted periods of time.

Keywords: Background modeling · deep learning · autoencoders

1 Introduction

Visual pieces of information such as images or video sequences are massively generated and used nowadays. Therefore, reliable and efficient ways to process that kind of data are needed more than ever. Video surveillance remains a very active field in the area of artificial vision, due to the fact that some demanding tasks have not been addressed adequately yet, as it is the case of background modeling, which consists of deciding whether an object of an image belongs to the scene foreground or background.

Robustness is a key feature which foreground detection algorithms must present. They should work continuously and they have to be prepared to cope with events which make the background characteristics vary. A change in the weather conditions in outdoor environments or lightning variations in indoor environments may compromise the reliability of moving object detection. Therefore, the algorithm performance must be kept at an acceptable level not only

for the initial video frames but also for the entire sequence. This goal is hard to achieve and many published methods stop working properly when changes in the environment occur.

Most of the foreground detection algorithms work at pixel level. They attempt to learn a model per pixel in order to compute the likelihood of each pixel to belong to one of the two possible classes: foreground or background. The main differences among the most referenced proposals reside in the underlying model that represents each pixel intensity of color over time. Wren et al. [14] defines pixel models based on a Gaussian distribution, whereas the GMM model [10] uses K distributions to manage multimodal funds. An intermediate approach can be found in Zivkovic [16], where as many Gaussians as necessary up to a maximum value (K) are considered. On the other hand, Elgammal et al. [2] uses kernel distributions to obtain non parametric probabilistic models. Finally, it must be cited SOBS [5] and FSOM [3], whose models are based on self-organizing maps, which are unsupervised neural networks in which a topology is defined. Model robustness is provided by combining each pixel output (probability of belonging to a moving object) with their neighbor ones.

In this work a foreground detection algorithm that attenuates the impact of noise in scene background modeling is presented. Each image will be divided into patches that are part of distinct shifted tilings of the video frame. As a consequence, each pixel will belong to different tiling patches. The noise that is present in the patches will be removed by a previously trained stacked autoencoder, which is an unsupervised deep learning neural network well suited to information representation, due to its ability to provide relevant data features [12]. Single layer autoencoders are proved to span the same subspace as a Principal Components Analysis technique [1]. The reduced patch information will be inputted to a mixture of Gaussian probabilistic model. Finally, each pixel classification will combine the classification outputs of the patches to which it belongs.

The paper is divided in the following sections: Section 2 presents the object detection methodology based on the analysis of image patches to obtain a foreground mask from an input frame; section 3 reports the experimental results over several public surveillance sequences and Section 4 concludes the article.

2 Methodology

Most previous approaches to background modeling in video sequences represent each pixel of the video frame separately. Our method intends to model patches of size $N \times N$ pixels, so that for each incoming video frame an estimation is made in order to know whether each patch belongs to the background of the scene. Furthermore, M shifted tilings of the video frame are considered, so that a particular pixel is classified by M background models. These M classifications are subsequently combined to yield a single classification output. The process is divided in two stages: firstly, a condensed representation of the patch, composed of significant features, is obtained by means of a previously trained Stacked

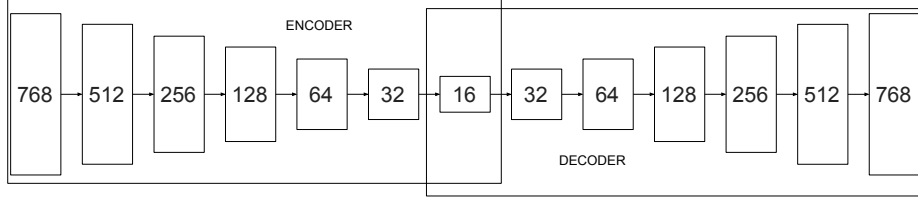


Fig. 1: Complete autoencoder structure with layers sizes.

Denoising Autoencoder (SDA) [12]; secondly, a probabilistic model classifies the patch according to their computed set of relevant features. Once all the patches have been classified the pixels are labelled accordingly.

2.1 Patch feature extraction

It turns out that stacked denoising autoencoders might find difficulties in modeling too small patches. Here we propose to overcome this limitation by using big $N \times N$ pixel patches, where N is big enough that the autoencoder models the patches adequately. Then we have M tilings of the video frame, so that each tiling is composed by $N \times N$ patches. The i -th tiling, where $i \in \{1, \dots, M\}$ is characterized by a unique shift vector $\mathbf{s}_i \in \{0, \dots, N-1\} \times \{0, \dots, N-1\}$, which makes it different from all the other tilings. Please note that the upper left corner of a $N \times N$ patch of the i -th tiling must be located at position \mathbf{s}_i in the video frame. The video frame is extended as required by symmetric (mirror) padding, so that all patches are complete with their $N \times N$ pixels irrespective of the shifts.

Let $\mathbf{X} \in \mathbb{R}^H$ be a patch of size $H = N^2$, where tristimulus pixel color values are assumed. A single stacked denoising autoencoder processes all the patches of all the tilings:

$$\tilde{\mathbf{X}} = g(f(\mathbf{X})) \quad (1)$$

$$f : \mathbb{R}^H \rightarrow \mathbb{R}^L \quad (2)$$

$$g : \mathbb{R}^L \rightarrow \mathbb{R}^H \quad (3)$$

where $\tilde{\mathbf{X}} \in \mathbb{R}^H$ is the reconstructed version of the input patch \mathbf{X} , f is the encoding part of the autoencoder, g is the decoding part of the autoencoder, and L is the number of neurons of the innermost layer of the neural architecture, i.e. the autoencoder reduces the high dimensional input of size H to a low dimensional set of features of size L with $L < H$.

The autoencoder is trained to minimize the reconstruction error \mathcal{E}_{train} :

$$\mathcal{E}_{train} = \sum_{i=1}^R \|\mathbf{X} - \tilde{\mathbf{X}}\|^2 \quad (4)$$

where R is the overall number of patches existing in the training data set, which is usually composed of video frames. Denoising autoencoders try to learn a robust representation made up of more general features which prevents from overtraining and diminishes the influence of scene factors such as illumination and local variation. In an attempt to enforce the invariance of the autoencoder to the diverse scene conditions, several authors [13][15] have used a training set that comprises a huge amount of generic natural image patches that may be corrupted instead of patches extracted from the frames corresponding to the video to process. This approach is followed in our proposal, where the training set for our single autoencoder is generated from the Tiny Images dataset [11].

2.2 Patch classification

As the video sequence progresses, the features which are discovered by the autoencoder are extracted, and a probabilistic model is learned for each patch of each tiling. This model aims to capture the main characteristics of the probability distribution of the feature vector $\mathbf{v} \in \mathbb{R}^L$:

$$\mathbf{v} = f(\mathbf{X}) \quad (5)$$

To this end, the mean $\mu_j = E[v_j]$ and the variance $\sigma_j^2 = E[(v_j - \mu_j)^2]$ of each component of \mathbf{v} are approximated by the Robbins-Monro stochastic approximation algorithm [7]. Initially, μ_j is set to the median reduced feature vector of the first video frames, while the initial value for σ_j^2 is obtained from the autoencoder training image set. During the training phase each probabilistic model characteristics are updated only if the patch j is classified as background:

$$\mu_{j,t+1} = (1 - \alpha) \mu_{j,t} + \alpha v_{j,t} \quad (6)$$

$$\sigma_{j,t+1}^2 = (1 - \alpha) \sigma_{j,t}^2 + \alpha (v_{j,t} - \mu_{j,t})^2 \quad (7)$$

where t is the time instant (the frame index) and α is the step size.

Each patch is declared to belong to the foreground whenever the number of components of the feature vector which are far from its estimated mean, as measured with respect to the estimated variance, is higher than a given threshold:

$$C < \sum_{j=1}^L \mathbb{I}(|v_{j,t} - \mu_{j,t}| > K \sigma_{j,t}) \quad (8)$$

where \mathbb{I} stands for the indicator function, C is a tunable parameter which specifies the number of components which must be far from its estimated mean to declare that the small patch belongs to the foreground, and K is another tunable parameter which specifies how many standard deviations an observation must depart from its estimated mean to be considered to be far away.

Each pixel of the video frame belongs to M patches, one per tiling. The fraction of these patches which have been declared as foreground is computed.

Table 1: Final parameter selection for each video. $\tau = 0.5$ in any case.

canoe	fountain01	fountain02
$C = 2, K = 1, \alpha = 0.001$	$C = 2, K = 2, \alpha = 0.01$	$C = 6, K = 0.5, \alpha = 0.001$
boats	pedestrians	overpass
$C = 6, K = 2, \alpha = 0.001$	$C = 10, K = 0.5, \alpha = 0.001$	$C = 3, K = 3, \alpha = 0.005$

Then the pixel is declared to belong to the foreground whenever the fraction is higher than a prespecified threshold τ , where $\tau \in [0, 1]$.

3 Experimental Results

3.1 Methods

Seven methods have been selected in order to make a performance comparison with our proposal: WrenGA [14], ZivkovicGMM [16], ElgammalKDE [2], SuB-SENSE [9], SC-SOBS ([6]), CL-VID [4] and FSOM [3].

The first four of these methods are available on BGS library [8]¹. SC-SOBS executable has been obtained from CVPRLAB web². FSOM and CL-VID code have been obtained from their authors' websites. The different method parameters are set to the default values indicated by the authors.

Our proposed approach has been implemented using Python. The neural network implementation makes use of the high-level application programming interface Keras³ which is based on TensorFlow⁴.

A thousand random images from Tiny Images dataset [11]⁵ have been used to train and test our autoencoder implementation. Total amount of autoencoder training data is 400,000 since each image has 32x32 pixels and we have divided each one to obtain four 16x16 images.

Input video sequences with added Gaussian noise have been generated once so all studied methods process the same sequences in order to be as fair as possible when comparing them. We do not use any additional post processing in any of the methods.

3.2 Sequences

A set of video sequences have been selected from the 2014 dataset of the ChangeDetection.net website⁶. Five of the selected scenes are from Dynamic Background

¹ <https://github.com/andrewssobral/bgslibrary>

² <http://cvprlab.uniparthenope.it/index.php/code/moving-object-detection-software-2.html>

³ <https://keras.io/>

⁴ <https://www.tensorflow.org/>

⁵ <http://groups.csail.mit.edu/vision/TinyImages/>

⁶ <http://changedetection.net/>

category, and another one from the Baseline one. *Fountain01* shows a road next to vertical water springs (432x288 pixels and 1184 frames), while in *Fountain02* a road next to a fountain spitting water out can be seen. (432x288 pixels and 1499 frames). *Canoe* presents a canoe going across a river with water and forest background (320x240 pixels and 1189 frames). *Boats* shows a river and a road. Various vehicles move on the road and various boats cross through the river. (320x240 pixels and 7999 frames). In *Overpass*, a road behind a bridge traversed by a man and a river is displayed (320x240 pixels and 3000 frames). Finally, *Pedestrians* is a baseline video where pedestrians walk over from one end of the screen to the other (360x240 pixels and 1099 frames).

3.3 Parameter selection

Five parameters must be fixed in order for our method to work properly (τ , M , C , K and α). τ has been set to 0.5 for all experiments, thus, a pixel is segmented as foreground if at least half of the M tiles where it belongs are considered as foreground. $M \in \{1, 4, 16, 64\}$ has been tested to study parameter M influence and figure 3 on page 8 shows that comparison without noise and with it. Our experiments reveal the greater M , the better performance. However we have selected $M = 16$ as our method version to compare with competitor methods so that a reasonable execution time is maintained. C , K , and α have been selected empirically based on our previous experience and preliminary experiments. Table 1 on page 5 shows final parameter selection for each video. The same configuration has been used for each noise and M value in experiments.










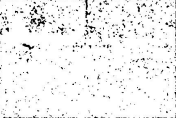



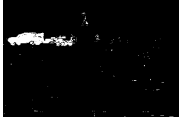
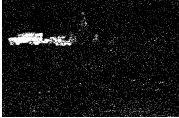
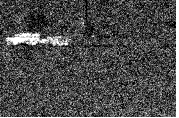


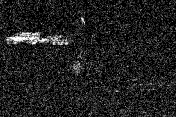








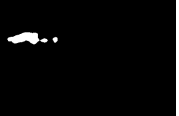
3.4 Evaluation

As a measure to perform a quantitative comparison among methods, the well-known *F-score* (also noted as *F-measure* or F_1 score) is used. It is defined as a balanced harmonic mean of precision and recall. This measure provides values in the interval $[0, 1]$, where values close to one mean better performance.

F-score has been calculated for each binary frame in Region of Interest (specified by ChangeDetection.net) generated using each previously mentioned method and we have obtained the mean for all frames with $TP + FN$ greater than zero.

Comparison among methods for videos with different Gaussian noise levels can be observed on table 2 on page 8. The table shows how the proposed method is able to deal with greater amount of noise than its contenders. While some of those methods can deal with Gaussian noise with $\sigma = 0.1$ (SUBSENSE, CL-VID and SC-SOBS, for example), their performance drops significantly in most tests where $\sigma = 0.2$. In figure 2 on page 7, it can be observed that our method copes with noise increasing faintly the number of *FN* pixels instead of increasing *FP* pixels.

Fig. 2: Qualitative results for frame 742 from fountain02 video. From left to right: images with different amount of Gaussian noise with mean 0. First row is original dataset input image with different amounts of Gaussian noise and ground-truth. Other rows correspond to foreground segmentation performed for various methods for each input image.

	$\sigma = 0$	$\sigma = 0.1$	$\sigma = 0.2$	GT
Sequence				
Ours				
FSOM				
KDE				
Wren				
Zivkovic				
CL-VID				
SC-SOBS				
SUBSENSE				

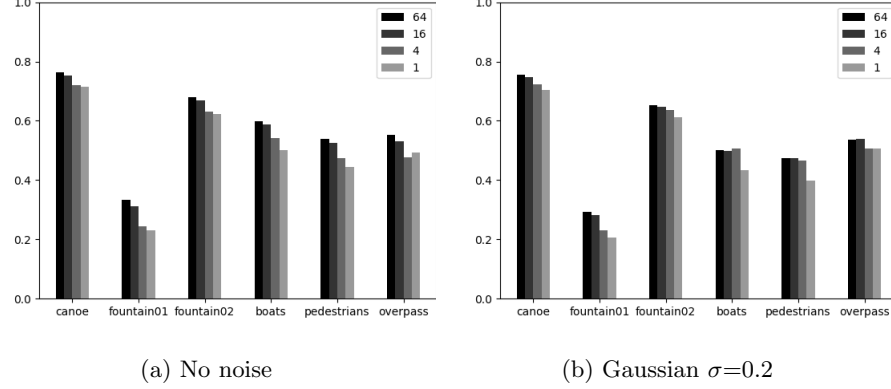


Fig. 3: Performance of our method using different number of tilings M . Sub-figure (a) shows results when there is no noise in videos. Sub-figure (b) shows performance when a Gaussian noise with $\sigma = 0.2$ is present. The greater M , the better average performance.

Table 2: Quantitative results. Each method $F1-Score$ is showed next to its ranking for each video and Gaussian noise level. Last row is the sum of all ranks where the lower, the better.

$\sigma = 0$								
	oursM=16	WREN	ZIVKOVIC	SC-SOBS	CL-VID	FSOM	KDE	SUBSENSE
canoe	0.7526 (4)	0.4584 (7)	0.6097 (6)	0.8178 (2)	0.8020 (3)	0.8234 (1)	0.2196 (8)	0.7142 (5)
fountain01	0.3124 (3)	0.1325 (7)	0.1767 (6)	0.3067 (4)	0.1823 (5)	0.4333 (2)	0.0429 (8)	0.7083 (1)
fountain02	0.6677 (4)	0.6053 (7)	0.6533 (6)	0.7886 (1)	0.6596 (5)	0.7516 (3)	0.1114 (8)	0.7777 (2)
boats	0.5870 (4)	0.3871 (7)	0.4803 (6)	0.7582 (1)	0.6910 (2)	0.6239 (3)	0.1359 (8)	0.5619 (5)
pedestrians	0.5250 (7)	0.7031 (3)	0.6813 (4)	0.7250 (1)	0.6748 (5)	0.6623 (6)	0.3557 (8)	0.7235 (2)
overpass	0.5320 (6)	0.4012 (7)	0.5470 (5)	0.6889 (3)	0.7237 (2)	0.7786 (1)	0.1595 (8)	0.6761 (4)
$\sigma = 0.1$								
	oursM=16	WREN	ZIVKOVIC	SC-SOBS	CL-VID	FSOM	KDE	SUBSENSE
canoe	0.7567 (1)	0.3600 (7)	0.5174 (5)	0.5868 (2)	0.5846 (3)	0.3828 (6)	0.1385 (8)	0.5420 (4)
fountain01	0.3214 (2)	0.0621 (7)	0.0811 (6)	0.1410 (3)	0.1192 (4)	0.1017 (5)	0.0114 (8)	0.5097 (1)
fountain02	0.6705 (2)	0.2376 (7)	0.3620 (6)	0.4595 (4)	0.5046 (3)	0.4043 (5)	0.0313 (8)	0.7724 (1)
boats	0.5836 (4)	0.2266 (7)	0.3897 (6)	0.5857 (3)	0.7771 (1)	0.6849 (2)	0.1072 (8)	0.4521 (5)
pedestrians	0.5104 (5)	0.4820 (6)	0.5984 (4)	0.6731 (3)	0.7013 (2)	0.4420 (7)	0.0995 (8)	0.7417 (1)
overpass	0.6048 (2)	0.3002 (6)	0.4723 (5)	0.4916 (4)	0.6823 (1)	0.5478 (3)	0.1024 (8)	0.2577 (7)
$\sigma = 0.2$								
	oursM=16	WREN	ZIVKOVIC	SC-SOBS	CL-VID	FSOM	KDE	SUBSENSE
canoe	0.7460 (1)	0.2285 (3)	0.3504 (2)	0.1826 (5)	0.1451 (6)	0.1293 (7)	0.1176 (8)	0.2004 (4)
fountain01	0.2825 (1)	0.0169 (4)	0.0254 (3)	0.0115 (5)	0.0093 (7)	0.0093 (7)	0.0089 (8)	0.0352 (2)
fountain02	0.6485 (1)	0.0538 (4)	0.0864 (3)	0.0306 (5)	0.0222 (8)	0.0234 (6)	0.0232 (7)	0.4946 (2)
boats	0.4983 (1)	0.1086 (4)	0.1685 (2)	0.0741 (5)	0.0737 (6)	0.1106 (3)	0.0732 (7)	0.0057 (8)
pedestrians	0.4737 (2)	0.1166 (4)	0.1766 (3)	0.0782 (5)	0.0466 (6)	0.0381 (8)	0.0454 (7)	0.7174 (1)
overpass	0.5383 (1)	0.1528 (3)	0.2559 (2)	0.1040 (6)	0.1228 (5)	0.1235 (4)	0.0848 (7)	0.0777 (8)
Σ rank	51	100	80	62	74	79	140	63

4 Conclusions

A methodology for detecting the foreground in video sequences has been presented. It combines M tilings of $N \times N$ patches of the video frame and a previously trained stacked autoencoder which attempts to discover significant features of the patches even in presence of noise. The reduced representation of each patch is provided to a multidimensional probabilistic model which determines the likelihood of a patch to belong to background or foreground.

The influence of the tilings in the model capability is clearly manifest. The higher the number of them, the better the performance. In the case of using two or more, they allow the model, which works at region level, to provide a particular pixel classification output which may differ from the classification output of the pixels of the same patch, because those pixels are also part of different patches in the remaining tilings. However, the computational cost inherent to the processing of patches of a new tiling must be taken into account. A trade-off between accuracy and computing time is needed and $M = 16$ is the recommended value according to the experiments.

Several heterogeneous scenes, with and without noise, have been processed and the results yielded by our method and other seven background modeling methods have been compared. According to those results, the method robustness must be highlighted. Not only is it able to keep a good performance even though noise appears but it is also the method that best works with very noisy sequences, which make the performance of the other methods fall drastically whereas the proposed method one is slightly diminished.

Acknowledgements

This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2014-53465-R, project name Video surveillance by active search of anomalous events, besides for the projects with codes TIN2016-75097-P and PPIT.UMA.B1.2017. It is also partially supported by the Autonomous Government of Andalusia (Spain) under grant TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga. They also gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan X GPUs used for this research. The authors would like to thank the grant of the Universidad de Malaga.

References

1. Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks* **2**(1), 53–58 (1989)

2. Elgammal, A., Harwood, D., Davis, L.: Non-parametric model for background subtraction. In: Computer Vision (ECCV). pp. 751–767. Springer (2000)
3. López-Rubio, E., Luque-Baena, R., Domínguez, E.: Foreground detection in video sequences with probabilistic self-organizing maps. *International Journal of Neural Systems* **21**(3), 225–246 (2011)
4. López-Rubio, E., Molina-Cabello, M.A., Luque-Baena, R.M., Domínguez, E.: Foreground detection by competitive learning for varying input distributions. *International Journal of Neural Systems* **28**(05), 1750056 (2018). <https://doi.org/10.1142/S0129065717500563>
5. Maddalena, L., Petrosino, A.: A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing* **17**(7), 1168–1177 (2008)
6. Maddalena, L., Petrosino, A.: The sobs algorithm: What are the limits? pp. 21–26 (06 2012)
7. Robbins, H., Monro, S.: A stochastic approximation method. *The Annals of Mathematical Statistics* **22**(3), 400–407 (1951)
8. Sobral, A., Bouwmans, T.: Bgs library: A library framework for algorithm’s evaluation in foreground/background segmentation. In: Background Modeling and Foreground Detection for Video Surveillance. CRC Press, Taylor and Francis (2014)
9. St-Charles, P., Bilodeau, G., Bergevin, R.: Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing* **24**(1), 359–373 (Jan 2015). <https://doi.org/10.1109/TIP.2014.2378053>
10. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. vol. 2, pp. 246–252 (1999)
11. Torralba, A., Fergus, R., Freeman, W.: 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **30**(11), 1958–1970 (2008)
12. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* **11**, 3371–3408 (2010)
13. Wang, N., Yeung, D.: Learning a deep compact image representation for visual tracking. In: Advances in Neural Inform. Processing Systems 26, pp. 809–817 (2013)
14. Wren, C., Azarbayejani, A., Darrell, T., Pentl, A.: Pfnder: Real-time tracking of the human body. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **19**(7), 780–785 (1997)
15. Zhang, Y., Li, X., Zhang, Z., Wu, F., Zhao, L.: Deep learning driven blockwise moving object detection with binary scene modeling. *Neurocomputing* **168**, 454 – 463 (2015)
16. Zivkovic, Z., van der Heijden, F.: Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters* **27**(7), 773–780 (2006)